Practical Work 1: Unit Testing

Huynh Vinh, Nam
huynh-vinh.nam@usth.edu.vn

Kieu Quoc, Viet
kieu-quoc.viet@usth.edu.vn

1 ShoppingCart Class Specification

The `ShoppingCart` class represents a virtual shopping cart for customers. It allows users to add items, update quantities, calculate the total price, and more. In this section, you are expected to develop a robust `ShoppingCart` class in Python.

1.1 Attributes

- `max_quantity`: total amount of items that the cart can hold.
- `items`: A dictionary store item data:
  - **Keys**: Unique item identifiers (product codes, generated automatically as "ITEM-X" where X is a sequential number).
  - **Values**: Dictionaries containing detailed information about each item:
    - `name`: The name of the item (string).
    - `quantity`: The number of items in the cart (int).
    - `price`: The price per item (float).

1.2 Methods

- `add_item(name, quantity, price)`: Adds a new item to the shopping cart or updates the quantity of an existing item.
  - **Parameters**:
    - `name`: The name of the item (string).
    - `quantity`: The number of item to add (int).
    - `price`: The price per item (float).
- `remove_item(name, quantity)`: Removes a quantity of items (or the whole item) from the shopping cart if it exists.

- `update_quantity(name, new_quantity)`: Updates the quantity of all the items in the cart.
  - **Parameters**:
    - name: The name of the item to update (string).
    - new_quantity: The new quantity to set (int).
- `view_cart()`: Displays the contents of the shopping cart in a formatted way.
- `checkout()`: Calculates and returns the total price of all items in the cart.

2 Testing Tasks

In this section, you are required to write test cases using the Python unit testing framework like **unittest** or **pytest** for test execution and reporting.

2.1 Test `view_cart()`:

- Test viewing an empty cart.
- Test viewing a cart with multiple items.
- Test the format of the returned string representation.

2.2 Test `add_item()`:

- Test adding an existing item to increase its quantity.
- Test adding an item when the cart is full
- Test adding an item with 0 quantity (should not be added).
- Test adding an item with negative price and negative quantity (should raise an exception).

2.3 Test `remove_item()`:

- Test removing an existing item from the cart.
- Test removing a quantity of existing item.
- Test removing an item that doesn't exist in the cart (should raise an exception).

2.4 Test `checkout()`:

- Test calculating the total with multiple items of different quantities and prices.
- Test calculating the total with an empty cart (should return 0).